PROGRAM-1

To implement addition and multiplication of 2D arrays.

ADDITION:-

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define row 10
#define col 10
int i,j;
int row1, col1;
int row2, col2;
float mat1[row][col];
float mat2[row][col];
float mat_res[row][col];
void mat_add(float mat1[row][col],int,int,float mat2[row][col],int,int,float
mat_res[row][col]);
void display(float mat[row][col],int,int); void input(float mat[row][col],int,int);
void mat_add(float mat1[row][col], int row1,int col1,float mat2[row][col],int row2, int col2, float
mat_res[row][col]);
{
int i,j;
if((row1==row2)&&(col1==col2))
{
printf("\nAddition is possible and Result is as follows\n");
for(i=0;i<row;i++)
for(j=0;j<col;j++)
mat_res[i][j]=mat1[i][j]+mat2[i][j];
display(mat_res,row1,col1);
}
else
printf("\nAddition is not possible\n");
exit(0);
}
void display(float mat[row][col],int r,int c)
{
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%f",mat[i][j]);
}
printf("\n");
} }
void input(float mat[row][col],int r,int c)
{
for(i=0;i<r;i++) {
for(j=0;j<c;j++) {
printf("Input value for:
%d:%d:",i+1,j+1);
scanf("%f",&mat[i][j]);
}
} } void main()
{
clrscr(); int row1,col1; int row2,col2;
float mat1[row][col];
float mat2[row][col];
float mat_res[row][col];
printf("\nInput the row of the matrix->1:");
scanf("%d",&row1);
printf("\nInput the col of the matrix->1:");
scanf("%d",&col1); printf("\nInput data for matrix-> 1\n");
input(mat1,row1,col1);
printf("\nInput the row of the matrix->2:");
scanf("%d",&row2);
printf("\nInput the col of the matrix->2:");
scanf("%d",&col2);
printf("\nInput data for matrix-> 2\n");
input(mat2,row2,col2);
printf("\nEntered Matrix First is as follows:\n");
display(mat1,row1,col1);
printf("\nEntered Matrix Two is as follows:\n");
display(mat2,row2,col2);
mat_add(mat1,row1,col1,mat2,row2,col2,mat_res);

}
```

OUTPUT:-

```
Input the row of the matrix->1:3
Input the col of the matrix->1:3
Input data for matrix->1
Input value for : 1 : 1: 11
Input value for : 1 : 2: 22
Input value for : 1 : 3: 33
Input value for : 2 : 1: 44
Input value for : 2 : 2: 55
Input value for : 2 : 3: 66
Input value for : 3 : 1: 77
Input value for : 3 : 2: 88
Input value for : 3 : 3: 99
Input the row of the matrix->2:3
Input the col of the matrix->2:3
Input data for matrix->2
Input value for : 1 : 1: 1
Input value for : 1 : 1: 2
Input value for : 1 : 1: 3
Input value for : 2 : 2: 4
Input value for : 2 : 2: 5
Input value for : 2 : 2: 6
Input value for : 3 : 3: 7
Input value for : 3 : 3: 8
Input value for : 3 : 3: 9 Entered Matrix First is as
follows:
11    22    33
44    55    66
77    88    99
Entered Matrix Two is as follows:
1    2    3
4    5    6
7    8    9
Addition is possible and Result is as follows:
12    24    36
48    60    72
84    96    108
```

MULTIPLICATION:-

```
include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define row 10
#define col 10
int i,j;
int row1, col1;
int row2, col2;
float mat1[row][col];
float mat2[row][col];
float mat_res[row][col];
void mat_mult(float mat1[row][col],int,int,float mat2[row][col],int,int,float
mat_res[row][col]);
void display(float mat[row][col],int,int);
void input(float mat[row][col],int,int);
void mat_mult(float mat1[row][col], int row1,int col1,float mat2[row][col],int row2, int col2, float
mat_res[row][col])
{
int i,j,k; if((col1==row2)
{
printf("\nMultiplication is possible and Result is as follows\n");
for(i=0;i<row1;i++)
for(j=0;j<col2;j++)
mat_res[i][j]=0; f
or(k=0;k<col1;k++)
{
mat_res[i][j]+=mat1[i][k]*mat2[k][j
} }
display(mat_res,row1,col2);
}
else
printf("\nMultiplication is not possible\n");
exit(0); }
void display(float mat[row][col],int r,int c)
{
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%f",mat[i][j]);
}
printf("\n");
} }
void input(float mat[row][col],int r,int c)
{
for(i=0;i<r;i++) {
for(j=0;j<c;j++) {
printf("Input value for: %d: %d:",i+1,j+1);
scanf("%f",&mat[i][j]);
}
} } void main()
{
clrscr(); int row1,col1;
int row2,col2;
float mat1[row][col];
float mat2[row][col];
float mat_res[row][col];
printf("\nInput the row of the matrix->1:");
scanf("%d",&row1);
printf("\nInput the col of the matrix->1:");
scanf("%d",&col1);
printf("\nInput data for matrix-> 1\n");
input(mat1,row1,col1);
printf("\nInput the row of the matrix->2:");
scanf("%d",&row2);
printf("\nInput the col of the matrix->2:");
scanf("%d",&col2);
printf("\nInput data for matrix-> 2\n");
input(mat2,row2,col2);
printf("\nEntered Matrix First is as follows:\n");
display(mat1,row1,col1);
printf("\nEntered Matrix Two is as follows:\n");
display(mat2,row2,col2);
mat_mult(mat1,row1,col1,mat2,row2,col2,mat_res);

}
```

OUTPUT:-

```
Input the row of the matrix->1:3
Input the col of the matrix->1:3
Input data for matrix->1
Input value for : 1 : 1: 1
Input value for : 1 : 2: 2
Input value for : 1 : 3: 3
Input value for : 2 : 1: 4
Input value for : 2 : 2: 5
Input value for : 2 : 3: 6
Input value for : 3 : 1: 7
Input value for : 3 : 2: 8
Input value for : 3 : 3: 9
Input the row of the matrix->2:3
Input the col of the matrix->2:3
Input data for matrix->2
Input value for : 1 : 1: 8
Input value for : 1 : 1: 9
Input value for : 1 : 1: 7
Input value for : 2 : 2: 6
Input value for : 2 : 2: 5
Input value for : 2 : 2: 4
Input value for : 3 : 3: 3
Input value for : 3 : 3: 2
Input value for : 3 : 3: 1
Entered Matrix First is as follows:
1    2    3
4    5    6
7    8     9
Entered Matrix Two is as follows:
8    9     7
6    5    4
3    2    1
Multiplication is possible and Result is as follows:
29    25    18
80    73    54
131 121    90
```

PROGRAM:-2

To implement transpose of matrix

```
#include<stdio.h>
int i,j; int value;
int mat[10][10];
void display(int,int);
void display_o(int transp[10][10],int,int);
void input(int transp[10][10],int,int);
void transpose(int transp[10][10],int,int);
void transpose(int transp[10][10],int row,int col)
{ for(i=0;i<row;i++) {
for(j=0;j<col;j++)
{
mat[i][j]=transp[j][i];
}
} }
void display(int row,int col)
{ for(i=0;i<row;i++) {
for(j=0;j<col;j++) {
printf("%d",mat[i][j]);
}
printf("\n");
} }
void display_o(int transp[10][10],int row,int col)
{ for(i=0;i<row;i++) {
for(j=0;j<col;j++) {
printf("%d",transp[i][j]);
}
printf("\n");
} }
void input(int transp[10][10],int row,int col)
{ for(i=0;i<row;i++) {
for(j=0;j<col;j++) {
printf("Input Value for: %d: %d:",i+1,j+1);
scanf("%d",&value);
transp[i][j]=value;
}
} }
void main() {
int row,col;
 int transp[10][10];
 printf("\nInput the number of rows:");
scanf("%d",&row);
printf("\nInput the number of cols:");
scanf("%d",&col);
input(transp,row,col);
 printf("\nEntered Matrix is as follows:\n");
display_o(transp,row,col);
transpose(transp,col,row);
printf("\nTranspose of above matrix is as follows:\n");
display(col,row); }
```

OUTPUT-

```
Input the number of rows: 3
Input the number if cols: 4
Input value for : 1 : 1: 1
Input value for : 1 : 2: 2
Input value for : 1 : 3: 3
Input value for : 1 : 4: 4
Input value for : 2 : 1: 5
Input value for : 2 : 2: 6
Input value for : 2 : 3: 7
Input value for : 2 : 4: 8
Input value for : 3 : 1: 9
Input value for : 3 : 2: 10
Input value for : 3 : 3: 11
Input value for : 3 : 4: 12
Entered Matrix is as follows:
1    2    3    4
5    6    7    8
9    10  11  12
Transpose of above Matrix is as follows:
1    5    9
2    6   10
3    7   11
4    8   12
```

PROGRAM-3

To implement stack using array.

```
#include <stdio.h>
#include<conio.h> #define MAXSIZE 5
struct stack {
 int stk[MAXSIZE];
int top;
};
typedef struct stack STACK; STACK s;
void push (void); int pop(void);
void display (void); void main () {
int choice;
int option = 1; clrscr ();
s.top = -1;
printf ("STACK OPERATION\n");
while (option) {
printf (" 1 PUSH \n");
printf (" 2 POP \n");
printf (" 3 DISPLAY \n");
printf (" 4 EXIT \n");
printf ("Enter your choice\n");
scanf ("%d", &choice);
switch (choice)
{
case 1: push();
break;
case 2: pop();
break;
case 3: display();
break;
case 4: return;
}
fflush (stdin);
printf ("Do you want to continue(Type 0 or 1)?\n");
scanf ("%d", &option);
} }
void push ()
{
int num;
if (s.top ==(MAXSIZE - 1))
{
 printf ("Stack isFull\n"); return;
}
else {
printf ("Enter the element to be pushed\n");
scanf ("%d", &num);
s.top = s.top + 1;
s.stk[s.top] = num; }
return; } int pop () {

int num; if (s.top == - 1) {
printf ("Stack is Empty\n");
return (s.top); }
else {
```

```
num = s.stk[s.top];
printf ("poped element is = %d\n", s.stk[s.top]);
s.top = s.top - 1;
}
 return(num);
}
 void display ()
{
int i; if (s.top == -1) {
 printf ("Stack is empty\n");
return; }
else
{
printf ("\nThe status of the stack is\n");
for (i = s.top; i >= 0; i--) {
printf ("%d\n", s.stk[i]);
}
}
printf ("\n");
}
```

OUTPUT:

```
STACK OPERATION
                    1 PUSH
                    2 POP
                    3 DISPLAY
                    4 EXIT
                    Enter your choice :
                    1
                    Enter the element to be Pushed
                    12
                    Do you want to continue (Type 0 or 1)?
                    1
                    1 PUSH
                    2 POP
                    3 DISPLAY
                    4 EXIT
                    Enter your choice :
                    1
                    Enter the element to be Pushed
                    32
                    Do you want to continue (Type 0 or 1)?
                    1
                    1 PUSH
                    2 POP
                    3 DISPLAY
                    4 EXIT
                    Enter your choice :
                    1
                    Enter the element to be pushed
```

```
24
Do you want to continue (Type 0 or 1)?
1
1 PUSH
2  POP
3 DISPLAY
4 EXIT
Enter your choice :
3
              The status of Stack is :
12
24
32
Do you want to continue (Type 0 or 1)?
1
1 PUSH
2 POP
3 DISPLAY
4 EXIT
Enter your choice :
2
Popped element is =12 Do you want to continue (Type
0 or 1)?
1
1 PUSH
2 POP
3 DISPLAY
4 EXIT
Enter your choice : 3
The status of Stack is :
24
32
  Do you want to continue (Type 0 or 1) 0
```

PROGRAM-4

To implement queue using array.

```
#include<stdio.h>
#include<conio.h>
#define MAX 50 int queue_array[MAX];
int rear=-1;
int front=-1;
void main() {
int choice;
while(1) {
printf("1. Insert element to queue :\n");
printf("2. Delete element from queue :\n");
printf("3. Display all elements of quque :\n");
printf("4. Quit :\n");
printf("Enter your choice:");
scanf("%d",&choice);
switch(choice) {
case 1:
insert();
break; case 2:
delete();
break; case 3:
display();
break;
case 4: exit(1);
default:
printf("Wrong choice :\n");
}
}
}
insert()
{ int add_item;
if(rear==MAX-1) printf("Queue Overflow :\n");
else { if(front==-1) front=0;
printf("Insert the element in queue    :\n ");
scanf("%d",&add_item); rear=rear+1;
queue_array[rear]= add_item;
}}
delete() {
if(front==-1 && front>rear)
{
printf("Queue Underflow :\n");
return;
}
else {
printf("Elements deleted from quque is : %d\n",queue_array[front]);
front= front+1;
}
}
display()
```

```
{
int i; if(front==-1)
printf("Queue is empty :\n\t");
else {
printf("Queue is :\n");
for(i=front;i<=rear;i++)
printf("%d",queue_array[i]);
printf("\n");
}
}
}
```

OUTPUT:-

```
MENU :
1.Insert into the queue
2.Delete from the queue
3.Display
4.Exit
Enter your choice :
1
Enter the item inserted :
45
After inserting queue is :
Node 1 : 45
Enter your choice :
25
Your choice is wrong
MENU :
   1.Insert into the queue
   2.Delete from the queue
   3.Display
   4.Exit
   Enter your choice : 3
   The queue is :
   Node 1: 45
   MENU :
   1.Insert into the queue
   2.Delete from the queue
   3.Display
   4.Exit
   Enter your choice :
   4
   Exit
```

PROGRAM-5

To implement circular queue using array.

```
#include<stdio.h>
#include<conio.h>
#define size 5 void insertq(int[], int);
void deleteq(int[]);
void display(int[]);
int front =- 1;
int rear =- 1;
int main() {
int n, ch; int queue[size];
do
{
printf("\n\n Circular Queue:\n1. Insert \n2. Delete\n3. Display\n0. Exit");
printf("\nEnter Choice 0-3? : ");
scanf("%d", &ch); switch (ch) {
case 1: printf("\nEnter number: ");
scanf("%d", &n); insertq(queue, n);
break;
case 2: deleteq(queue);
break;
case 3: display(queue);
break; }while (ch != 0);
}
void insertq(int queue[], int item)
if ((front == 0 && rear == size - 1) || (front == rear + 1))
{
printf("queue is full");
  return;
  }
else if (rear ==    - 1)
{
rear++;
front++;
}
else if (rear == size - 1 && front > 0)
{
rear = 0;
}
else {
rear++;
}
queue[rear] = item;
}
void display(int queue[])
{
int i;
 printf("\n");
if (front > rear)
{
```

```
for (i = front; i < size; i++)
{
printf("%d ", queue[i]);
}
for (i = 0; i <= rear; i++)
printf("%d ", queue[i]);
}
else
 {
for (i = front; i <= rear; i++)
printf("%d ", queue[i]);
} }

void deleteq(int queue[])
{
if (front ==      - 1)
{
printf("Queue is empty ");
}
else if (front == rear)
{
printf("\n %d deleted", queue[front]);
front = - 1;
rear =   - 1;
}
else {
printf("\n %d deleted", queue[front]);
front++;
}
```

OUTPUT-

```
Circular Queue :
1.Insert
2.Delete
3.Display
0.Exit
Enter choice 0-3 ? 1
Enter number : 20
Circular Queue :
1.Insert
2.Delete
3.Display
0.Exit
Enter choice 0-3 ? 1
Enter number : 40
Enter number : 20
Circular Queue :
1.Insert
2.Delete
3.Display
0.Exit
Enter choice 0-3 ?
3
20 40
Circular Queue :
1.Insert
2.Delete
3.Display 0.Exit
Enter choice 0-3 ?
```

PROGRAM-6

To implement stack using linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct Node { int data;
struct Node *next;
}
*top = NULL; // Initially the list is empty
void push(int);
void pop();
void display();
int main() {
 int choice, value;
printf("\nIMPLEMENTING STACKS USING LINKED LISTS\n");
while(1){
printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");
printf("\nEnter your choice : ");
scanf("%d",&choice); switch(choice)
{
case 1: printf("\nEnter the value to insert: ");
scanf("%d", &value);
push(value);
break;
case 2: pop();
break;
case 3: display();
break;
case 4: exit(0);
break;
default:
printf("\nInvalid Choice\n");
}}}
void push(int value)
{
struct Node *newNode;
newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value; // get value for the node
if(top == NULL)
newNode->next = NULL;
else
newNode->next = top; // Make the node as TOP top = newNode;
printf("Node is Inserted\n\n");
}
void pop()
{
if(top == NULL)
printf("\nEMPTY STACK\n");
else{
struct Node *temp = top;
printf("\nPopped Element : %d", temp->data);
printf("\n");
top = temp->next; // After popping, make the next node as TOP
free(temp); }}
void display()
{
if(top == NULL)
printf("\nEMPTY STACK\n");
else {
printf("The stack is \n");
struct Node *temp = top;
while(temp->next != NULL){
 printf("%d--->",temp->data);
temp = temp -> next;
}
printf("%d--->NULL\n\n",temp->data);
}}
```

OUTPUT:-

```
IMPLEMENTING STACK USING ARRAY
1.  Push
2.  Pop
3.  Display
4.  Exit
Enter your choice : 1
Enter the value to insert : 15
Node is inserted
1.  Push
2.  Pop
3.  Display
4.  Exit
Enter your choice : 1
Enter the value to insert : 30
Node is inserted
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 3
The stack is
30---->15--->NULL
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 2 Popped elements :15
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 2
STACK UNDERFLOW
1.Push
2.Pop
3.Display
4.Exit
Enter your choice : 4
```

PROGRAM-7

To implement of queue using linked list.

```
#include<stdio.h>

#include<conio.h>

struct Node

{

int data;

struct Node *next;

}
*front = NULL,*rear = NULL;
void insert(int);

void delete();

void display();

void main()
{

 int choice, value;

      clrscr();

      printf("\n:: Queue Implementation using Linked List ::\n");

      while(1){

      printf("\n****** MENU ******\n");

      printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");

      printf("Enter your choice: ");

      scanf("%d",&choice);

      switch(choice){

                 case 1: printf("Enter the value to be insert: ");

                 scanf("%d", &value);

                 insert(value);

                 break;

                 case 2: delete();

                 break;

                 case 3: display();

                 break;

                 case 4: exit(0);
```

```
            default: printf("\nWrong selection!!! Please try again!!!\n");
            }
    } }
    void insert(int value)
{
struct Node *newNode; newNode = (structNode*)malloc(sizeof(structNode));
newNode->data = value;
newNode -> next = NULL;
if(front == NULL)
        front = rear = newNode;
    else{
        rear -> next = newNode; rear = newNode;
    }
    printf("\nInsertion is Success!!!\n");
}
void delete()
{
if(front == NULL)
        printf("\nQueue is Empty!!!\n");
    else{
        struct Node *temp = front;
        front = front -> next;
        printf("\nDeleted element: %d\n", temp->data);
        free(temp);
    } }
    void display()
{
if(front == NULL)
        printf("\nQueue is Empty!!!\n"); else{
        struct Node *temp = front;
        while(temp->next != NULL){
            printf("%d--->",temp->data);
                      temp = temp -> next;
        }
        printf("%d--->NULL\n",temp->data);
    }
}
```

OUTPUT-

```
Queue Implementation using Linked List ::
****** MENU ******
1.  Insert
2.  Delete
3.  Display 4. Exit
Enter your choice:1
Enter the value to be insert:10 Insertion is Success!!!
****** MENU *****
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:1
Enter the value to be insert:20
10--->20--->NULL
****** MENU *****
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:2
Deleted element:10
****** MENU *****
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:3
20--->NULL
```

```
****** MENU *****
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:4
```

PROGRAM-8

To implement BFS using linked list.

```
#include<stdio.h>

#include<conio.h>

int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;

void bfs(int v) {

for(i = 1; i <= n; i++)

if(a[v][i] && !visited[i]) q[++r] = i;

if(f <= r) {

visited[q[f]] = 1;

bfs(q[f++]);

} }

void main() {

clrscr();

int v;

printf("Enter the number of vertices: ");

scanf("%d",&n);

for(i=1; i <= n; i++) {

q[i] = 0;

visited[i] = 0;

}

printf("\nEnter graph data in matrix form:\n");

for(i=1; i<=n; i++) {

for(j=1;j<=n;j++) {

scanf("%d", &a[i][j]);

} }

printf("Enter the starting vertex: ");

scanf("%d", &v); bfs(v);

printf("\nThe node which are reachable are:");
for(i=1; i <= n; i++) {
if(visited[i])
printf(" %d", i);
else {
printf("\nBFS is not possible. All nodes are not reachable!");
break; }
}
getch();
}
```

OUTPUT:-

```
Enter the number of vertices:3
Enter graph data in matrix form:
2 4 5 2 3 4 1 7 8
Enter the starting vertex:2
The node which are reachable are:1 2 3
```

PROGRAM-9

To implement DFS using linked list.

```
#include<stdio.h>
#include<stdlib.h>
#define scan(a) scanf("%d", &a)
#define print(a) printf("%d", a)
#define nline printf("\n")
#define fl(i,a,b) for(i=a; i<b; i++)
#define rev(i,a,b) for(i=a; i>=b; i--) #define sspace
printf(" ") typedef struct listnode
{ int data; struct listnode* next;
}listnode; typedef struct list {
listnode *head;
}list;
typedef struct graph
{
int vertices; list* array;
}
graph;
int visited[1000];
graph* creategraph(int n)
{
int i;
graph* G=(graph *)(malloc(sizeof(graph)));
G->vertices = n;
G->array = (list *)malloc(n * sizeof(struct list));
fl(i,0,n)
G->array[i].head = NULL; return G;
}
void addedge(graph* G, int src, int dest)
{
listnode* newnode;
newnode=(listnode *)(malloc(sizeof(listnode)));
newnode->data=dest; newnode->next=G->array[src].head;
G->array[src].head=newnode;
}
void traverse(graph* G, int n)
{
graph* temp=G;
int i; list* temp_list; listnode*
temp_node;
fl(i,0,n)
{ temp_node=temp->array[i].head;
while(temp_node!=NULL)
{
printf("%d -> %d\t", i, temp_node->data);
temp_node=temp_node->next;
}
nline;
}
return;
}
void dfs(graph* G, int v)
{
int i, j;
printf("%d ", v);
visited[v]=1;
listnode* temp_node=G->array[v].head;
while(temp_node!=NULL)
{
if(!visited[temp_node->data])
{
dfs(G,temp_node->data);
}
temp_node=temp_node->next;
}
return;
}
int main() {
int n, i, j, k, temp, m, a, b;
printf("Enter the number of nodes : ");
scan(n);
printf("Enter the number of edges : ");
scan(m);
printf("Enter the edges : ");
nline;
int ini=n+1;
graph* G=creategraph(n+1);
fl(i,0,m) {
scan(a);
scan(b);
addedge(G,a,b);
if(a<ini)
int=a;
}
printf("Traversing the adjacency list : ");
nline;
traverse(G,n);
nline;
printf("DFS traversal starting with %d : ", ini);
nline;
dfs(G,ini);
nline;
return 0;
}
```

OUTPUT:-

```
Enter the number of nodes : 9
Enter the number of edges :15
Enter the edges :
0 1
1  3
2  2
1 4
1  5
2  3
2  5
3  6
4  7
5  6
5 7
5  8
6  9
7  8
8  9
Traversing the adjacency list :
0->3 0->1
1->4 1->5 1->2
2->5 2->3
3->6
4->7
5->6 5->7 5->8
6->9
7->8
8->9
DFS traversal starting with:0
0 3 6 9 1 4 7 8 5 2
```

PROGRAM-10

To Implement Linear Search.

```
#include<stdio.h>

#include<conio.h>

void main() {

int arr[10];

int i,num,a,found=0;

clrscr();

printf("Enter the value of num\n");

scanf("%d",&num);

printf("Enter the elements one by one\n");

for(i=0;i<num;i++) {

scanf("%d",&arr[i]);

}

 printf("Input array is\n");

 for(i=0;i<num;i++) {

printf("%d\n",arr[i]);

}

printf("Enter the elements to be searched\n");

scanf("%d",&a); for(i=0;i<num;i++) {

if(a==arr[i])
{ found=1;

break;

} } if(found==1)

 printf("Element is present in the array \n");

else

printf("Element is not present in the array \n");
getch();

}
```

OUTPUT-

```
Enter the value of num
4
Enter the elements one by one :-
2
7
9
5
Input array is :2
7
9
5
Enter the number to be searched:7
Element is present in the array
```

PROGRAM-11

To implement Binary Search.

```
#include<stdio.h>

#include<conio.h>

void main() {

int c,first,last,middle,n,search,arr[100];

clrscr();

printf("Enter numbers of elements :\n");

scanf("%d",&n);

printf("Enter %d integer \n",n);

 for(c=0;c<n;c++)

 scanf("%d",&arr[c]);

printf("Enter value to find \n");

 scanf("%d",&search);

 first=0;

last=n-1;

 middle=(first+last)/2;

while(first<=last)

 {

if(arr[middle]<search) first=middle+1;

 else

if(arr[middle]==search)

{

printf("%d Found at location \n",search,first);

break;

}
else last=middle-1;

middle=(first+last)/2;

}


if(first>last)

printf("Not found ! %d don't present in the list\n",search);

getch();

}
```

OUTPUT-

```
Enter the numbers of elements:4
Enter 4 integers :26
87
54
22
Enter value to find out:-
87
87 Found at location
Enter the numbers of elements:4
Enter 4 integers:34
59
85
20
Enter value to find out:-
95
Not Found! 95 don’t present in the list
```

PROGRAM-13

To implement Bubble Sort.

```
#include<stdio.h>
int main() {
 int array[100], n, i, j, swap;
printf("Enter number of elementsn");
scanf("%d", &n);
printf("Enter %d Numbers:n", n);
for(i = 0; i < n; i++)
scanf("%d", &array[i]);
for(i = 0 ; i < n - 1; i++) {
for(j = 0 ; j < n-i-1; j++)
{
if(array[j] > array[j+1]) {
 swap=array[j];
array[j]=array[j+1];
array[j+1]=swap;
}
} }
printf("Sorted Array:n");
for(i = 0; i < n; i++)
printf("%dn", array[i]);
return 0;
}
```

OUTPUT-

```
Enter number of elements: 5
Enter 5 Numbers:
7
4
2
9
15 Sorted Array:
2
4
7
9
15
```

PROGRAM-13

To implement Selection Sort

```
#include<stdio.h>
#include<conio.h>
void main() {
int arr[100],n,c,d,position,swap;
clrscr();
printf("Enter number of elements :-\n");
scanf("%d",&n);
printf("Enter %d integers :-\n",n);
for(c=0;c<n;c++)
scanf("%d",&arr[c]);
for(c=0;c<(n-1);c++)
{
position=c;
for(d=c+1;d<n;d++) {
if(arr[position]>arr[d]) position=d;
}
if(position!=c) {
 swap=arr[c];
arr[c]=arr[position];
arr[position]=swap;
}
printf("Sorted List In Ascending Order :-\n");
for(c=0;c<n;c++)
printf("%d\n",arr[c]);
getch();

}
```

OUTPUT:

```
Enter total number of elements;5
Enter 5 elements:55
96
23
48
2
Sorted list in Ascending Order:2
23
48
55
96
```

PROGRAM-14

To Implement Insertion Sort

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,num,temp,arr[20];
clrscr();
printf("Enter total elements :\n");
scanf("%d",&num);
printf("Enter %d elements :\n",num);
for(i=0;i<num;i++) {
scanf("%d",&arr[i]);
}
for(i=1;i<num;i++)
 {
 temp=arr[i];
j=i-1;
while(temp<arr[j]&&(j>=0))
{
arr[j+1]=arr[j];
j=j-1; } arr[j+1]=temp;
}
printf("After Sorting :\n");
for(i=0;i<num;i++) {
printf("%d\n",arr[i]);
}
getch();

 }
```

OUTPUT:-

```
Enter total numbers of elements:-5 Enter 5 elements:
12
6
15
3
9
After sorting:3
6
9
12
15
```

PROGRAM-15

To implement Merge Sort

```
#include<stdio.h>
#include<conio.h>
#define max 10
int a[11]={10,14,19,26,27,31,33,35,42,45,1};
int b[10];
void merging(int low,int mid,int high)
{
int l1,l2,i;
clrscr();
for(l1=low,l2=mid+1,i=low;l1<=mid&&l2<=high;i++)
{
if(a[l1]<=a[l2]) b[i]=a[l1++];
else b[i]=a[l2++];
}
while(l1<=mid) b[i++]=a[l1++];
while(l2<=high) b[i++]=a[l2++];
for(i=low;i<=high;i++) a[i]=b[i];
}
void sort(int low,int high)
{
 int mid;
if(low<high)
{
 mid=(low+high)/2;
sort(low,mid);
sort(mid+1,high);
merging(low,mid,high);
}
else {
return;
} }
void main()
{
int i;
printf("List Before Sorting :\n");
for(i=0;i<=max;i++)
printf("\t%d",a[i]);
sort(0,max);
printf("\nList After Sorting :\n");
for(i=0;i<=max;i++)
printf("\n%d",a[i]);

getch();

}
```

OUTPUT:-

```
List before Sorting:10
                    14
                     19
                    26
                    27
                    31
                    33
                    35
                    42
                    45
                    1
List After Sorting:1
                    10
                    14
                    19
                    26
                    27
                    31
                    33
                    35
                    42
                    45
```

PROGRAM-16

To Implement Heap Sort.

```
#include<stdio.h>
#include<conio.h>
void create(int[]);
void down_adjust(int[],int);
void main() {
int n,i,heap[30],last,temp;
clrscr();
printf("\nEnter number of elements :-");
scanf("%d",&n);
printf("\nEnter the elements :-\n");
for(i=1;i<=n;i++)
scanf("%d",&heap[i]);
heap[0]=n;
create(heap);
while(heap[0]>1) {
last=heap[0];
temp=heap[1];
heap[1]=heap[last];
heap[last]=temp; heap[0]--;
down_adjust(heap,1);
}
printf("Array after Sorting :-\n");
for(i=1;i<=n;i++)
printf("%d",heap[i]);
getch();
}
void create(int heap[])
{
int i,n;
n=heap[0];
for(i=n/2;i>=1;i--)
down_adjust(heap,i);
}
void down_adjust(int heap[],int i)
{
int j,n,temp,flag=1;
n=heap[0];
while(2*i<=n&&flag==1)
{
j=2*i;
if(j+1<=n&&heap[j+1]>heap[j])
j=j+1;
if(heap[i]>heap[j])
flag=0;
else {
temp=heap[i];
heap[i]=heap[j];
heap[j]=temp;
i=j;
}
}
}
```

OUTPUT:-

```
Enter number of elements:-
5
Enter the elements;12
8
11
27
30
After Sorting:-
8
11
12
27
30
```

PROGRAM-17

To implement Matrix Multiplication by Strassen’s algorithm.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int a[2][2],b[2][2],c[2][2],i,j;
int m1,m2,m3,m4,m5,m6,m7;
printf("Enter the 4 elements of first matrix: ");
for(i=0;i<2;i++)
for(j=0;j<2;j++)
scanf("%d",&a[i][j]);
printf("Enter the 4 elements of second matrix: ");
for(i=0;i<2;i++)
for(j=0;j<2;j++)
scanf("%d",&b[i][j]);
printf("\nThe first matrix is\n");
for(i=0;i<2;i++) {
printf("\n"); for(j=0;j<2;j++)
printf("%d\t",a[i][j]);
}
printf("\nThe second matrix is\n");
for(i=0;i<2;i++) {
printf("\n");
for(j=0;j<2;j++)
printf("%d\t",b[i][j]); }
m1= (a[0][0] + a[1][1])*(b[0][0]+b[1][1]);
m2= (a[1][0]+a[1][1])*b[0][0];
m3= a[0][0]*(b[0][1]-b[1][1]);
m4= a[1][1]*(b[1][0]-b[0][0]);
m5= (a[0][0]+a[0][1])*b[1][1];
m6= (a[1][0]-a[0][0])*(b[0][0]+b[0][1]);
m7= (a[0][1]-a[1][1])*(b[1][0]+b[1][1]);
c[0][0]=m1+m4-m5+m7;
c[0][1]=m3+m5;
c[1][0]=m2+m4;
c[1][1]=m1-m2+m3+m6;
printf("\nAfter multiplication using \n");
for(i=0;i<2;i++) {
printf("\n");
for(j=0;j<2;j++)
printf("%d\t",c[i][j]);
}
return 0;
}
```

OUTPUT-

```
Enter the 4 elements of first matrix:
```

```
1 2
3 4
Enter the 4 elements of second matrix:
5 6
7 8
The first matrix is:
1    2
3    4
The second matrix is:
5    6
7    8
After multiplication using :
19    22
43    50
```

PROGRAM-18

Find Minimum Spanning Tree using Kruskal’s Algorithm.

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <algorithm> using namespace std;
struct Edge {
int src, dest, weight;
};
struct compare {
bool operator() (Edge const &a, Edge const &b) const {
return a.weight > b.weight;
}
};
class DisjointSet
{ unordered_map<int, int> parent;
public:
  void makeSet(int N)
{
for (int i = 0; i < N; i++) {
parent[i] = i;
} }
int Find(int k) {
if (parent[k] == k) {
return k;
}
return Find(parent[k]);
}
void Union(int a, int b)
{
int x = Find(a);
int y = Find(b);
parent[x] = y;
}
};
vector<Edge> kruskalAlgo(vector<Edge> edges, int N)
{
vector<Edge> MST;
DisjointSet ds;
ds.makeSet(N);
sort(edges.begin(), edges.end(), compare());
while (MST.size() != N - 1)
{
Edge next_edge = edges.back();
edges.pop_back();
int x = ds.Find(next_edge.src);
int y = ds.Find(next_edge.dest);
if (x != y)
{
MST.push_back(next_edge);
ds.Union(x, y);
} }
return MST;
}
int main() {
vector<Edge> edges = {
{ 0, 1, 7 }, { 1, 2, 8 }, { 0, 3, 5 }, { 1, 3, 9 },
{ 1, 4, 7 }, { 2, 4, 5 }, { 3, 4, 15 }, { 3, 5, 6 },
{ 4, 5, 8 }, { 4, 6, 9 }, { 5, 6, 11 }};
int N = 7;
vector<Edge> e = kruskalAlgo(edges, N);
for (Edge &edge: e) {
cout << "(" << edge.src << ", " << edge.dest << ", "<< edge.weight << ")" << endl;
}
return 0;
}
```

OUTPUT:-

```
(2, 4, 5)
(0, 3, 5)
(3, 5, 6)
(1, 4, 7)
(0, 1, 7)
(4, 5, 8)
```